Sections and Chapters

Gubert Farnsworth

Ngày 14 tháng 12 năm 2020

STRING MATCHING

1 String Matching Problem

Tìm kiếm chuỗi hiện diện rất nhiều trong cuộc sống, ví dụ như: tìm tên thầy dạy DSA trong danh sách giảng viên, tìm tên trong bảng điểm,.. hoặc trong khoa học, như là tìm liệu cấu trúc ADN của virus này có trong virus khác hay không.
Trong Tin học, các trình soạn thảo văn bản thường phải tìm kiếm (tất cả) các lần xuất hiện của một đoạn văn bản trong một văn bản dài. Thông thường văn bản được chỉnh sửa liên tục, và các phần văn bản cần tìm kiếm thì được nhập bởi người dùng. Việc phát minh ra các thuật toán tìm kiếm chuỗi hiệu quả đã giúp ích rất nhiều cho các bài toán kể trên.

Bài toán tìm kiếm chuỗi thường được mô tả theo mô hình sau:
Một chuỗi cần tìm kiếm S có dạng một chuỗi kí tự S[1..m], cần tìm chuỗi S trong một đoạn văn bản T[1..n], với m <= n. Đồng thời, tất cả các ký tự trong T và S đều thuộc về một tập hữu hạn các ký tự cho trước. Chuỗi S được gọi là xuất hiện bắt đầu từ vị trí p + 1 nếu thoả điều kiện: T[p + j] = S[j], với 1 <= j <= m.
Có 3 thuật toán thường được dùng để tìm kiếm chuỗi:

- Thuật toán Brute-force (vét cạn, hay còn gọi là thuật trâu)
- Rabin - Karp
- Knuth - Morris - Pratt

Chi tiết của từng thuật toán sẽ được trình bày bên dưới.

2 String Matching Algorithms

1. Brute-force

   Giải thuật Brute-Force, hay còn gọi là vét cạn, là thuật toán đơn giản nhất trong các thuật toán tìm kiếm chuỗi con *pattern* trong chuỗi cha *text*.

   Có thể giải thích đơn giản, giải thuật Brute-Force so sánh lần lượt mỗi chuỗi con *subtext* của *text* có cùng chiều dài với *pattern* với *pattern*, nếu tìm được, trả về kết quả là vị trí được tìm thấy; khi không tìm được kết quả mong muốn, trả về giá trị quy ước là không tìm thấy.

   Trong ví dụ sau, ta sẽ làm rõ cách hoạt động của giải thuật này:
   text = `Let them go!`
   pattern = `them`

Let them go!
them

Let them go!
 them

Let them go!
  them

Let them go!
them

Let them go!
them

Let them go!
them

Let them go!
them

Let them go!
them

Let them go!
them

Ta tìm thấy chuỗi pattern tại vị trí thứ 4!

Từ ví dụ trên, ta thiết kế mã giả cho giải thuật Brute-Force:

```
        vị trí tìm thấy = -1
        subtext = chuỗi con đầu text có độ dài bằng pattern
        while (chưa tìm thấy hoặc chưa tới cuối text)
          if (từng ký tự của subtext = pattern):
            trả về vị trí
          else:
            dịch chuyển chuỗi con subtext trong text sang phải 1 chữ cái
        Trả về: vị trí tìm thấy
```

Với chuỗi *pattern* có độ dài là M, chuỗi *text* có độ dài là N
Phân tích độ phức tạp của thuật toán trong trường hợp xấu nhất:

- Mỗi lần so sánh với *subtext*, *pattern* phải so sánh nhiều nhất là M lần (trong trường hợp cả M - 1 ký tự đầu đều đúng).
- Có tất cả N - M + 1 chuỗi, vậy số chuỗi cần so sánh nhiều nhất là N - M + 1 *subtext* như vậy (trong trường hợp N - M + 2 chuỗi *subext* đầu không trùng với *pattern*)
  → Cần M(N - M + 1) lần. Vì duyệt tới cuối mảng nên đây là trường hợp tìm thấy ở cuối mảng, hoặc không tìm thấy
  → Cận trên $O(MN)$ (vì N $>$ N - M + 1).
  → Cấp phát bộ nhớ: 0.

Phân tích độ phức tạp của thuật toán trong trường hợp tốt nhất:

- Trong trường hợp tốt nhất, có thể thấy *pattern* chính là *subtext* đầu tiên của *text*.
- Như vậy, chỉ cần tốn M lần so sánh các ký tự.
  → Cần M lần.
  → Cận trên $O(M)$.
  → Cấp phát bộ nhớ: 0.

Đánh giá:

- Dễ hiểu, thuật toán này chỉ duyệt từ đầu đến cuối, so sánh tuần tự từng chuỗi con với chuỗi cần tìm kiếm.
- Không cần bước tiền xử lý (như các thuật toán được trình bày bên dưới).
- Độ phức tạp $O(MN)$. Không cần xin thêm bộ nhớ.

PROGRAMING

(abc)

1 Introduce

2 Example Test